EATON CORPORATE

Eaton Industries SAS
GreenDataNet - System Architecture
Arnaud Quette, Gerald Guillaume
30/10/2014
Rev 0.4

1. TABLE OF CONTENTS

2. REVISION SHEET

Revision Number	Date	Brief summary of changes
V0.1	17/07/2014	Initial System Architecture proposition
V0.2	12/09/2014	General completion
V0.3	28/10/2014	General completion
V0.3	30/10/2014	Update General architecture diagram

3. INTRODUCTION

3.1. DOCUMENT PURPOSE

The intent of this document is to provide a design specification of GreenDataNet Data Centre Energy Monitoring and Optimisation Software.

3.2. DEFINITION, ACRONYMS AND ABBREVATIONS

3.2.1. KEY ACRONYMS AND ABBREVATIONS

AC	Alternating Current
AEMS	Aggregated Energy Management System
AMQP	Advanced Message Queuing Protocol Standard
BMS	Battery Management System
CER	Cooling Efficiency Ration
CPU	Central Processing Unit
CUE	Carbon Usage Effectiveness
DC	Direct Current
DCIM	Data Centre Infrastructure Management
DOW	Description of Work
EC	European Commission
EPA	Environmental Protection Agency
ERE	Energy Reuse Effectiveness
EV	Electric Vehicle
GC	GreenDataNet Controller
GHG	Greenhouse Gas
HVAC	Heating, Ventilation and Air Conditioning
IAB	Industry Advisory Board
ICT	Information and Communication Technology
IPR	Intellectual Property Rights
IT	Information Technology
ITEE	IT equipment Energy Efficiency
KPI	Key Performance Indicator
LEED	Leadership in Energy and Environmental Design
MoM	Message oriented Middleware
NIC	Network Interface Controller
NoC	Network-on-Chip
OOB	Out Of Band (control channel)
PCC	Project Coordination Committee
PDU	Power Distribution Unit

PSB	Project Steering Board
PUE	Power Usage Effectiveness
PV	Photovoltaic
RC	Rack Controller, synonym of GC – GreenDataNet Controller
REF	Renewable Energy Factor
SDK	Software Development Kit
SEMS	Smart Energy Management System
SLA	Service Level Agreement
SME	Small or Medium size Enterprise
SOA	Service Oriented Architecture
SoC	System-on-Chip
UPS	Uninterruptible Power Supply
WP	Work Package
WUE	Water Usage Effectiveness

3.1. LICENCE TERMS

The core software solution will be implemented as an open-source platform, to allow third parties and GreenDataNet partners to provide additional optimisation modules and ensure the long-term sustainability of the project.

4. SYSTEM ARCHITECTURE OVERVIEW

4.1. INTRODUCTION

Considering the Urban Data Centre specification and the Survey of Key emerging IT Trends, the present functional architecture will provide detailed visibility on every level, or layer, of the data centre, including:

- The devices located in racks (servers, storage, network, power,…)
- The racks that form a datacenter,
- Every grouping level in-between, such as rows and rooms.

Moreover, the system must address:

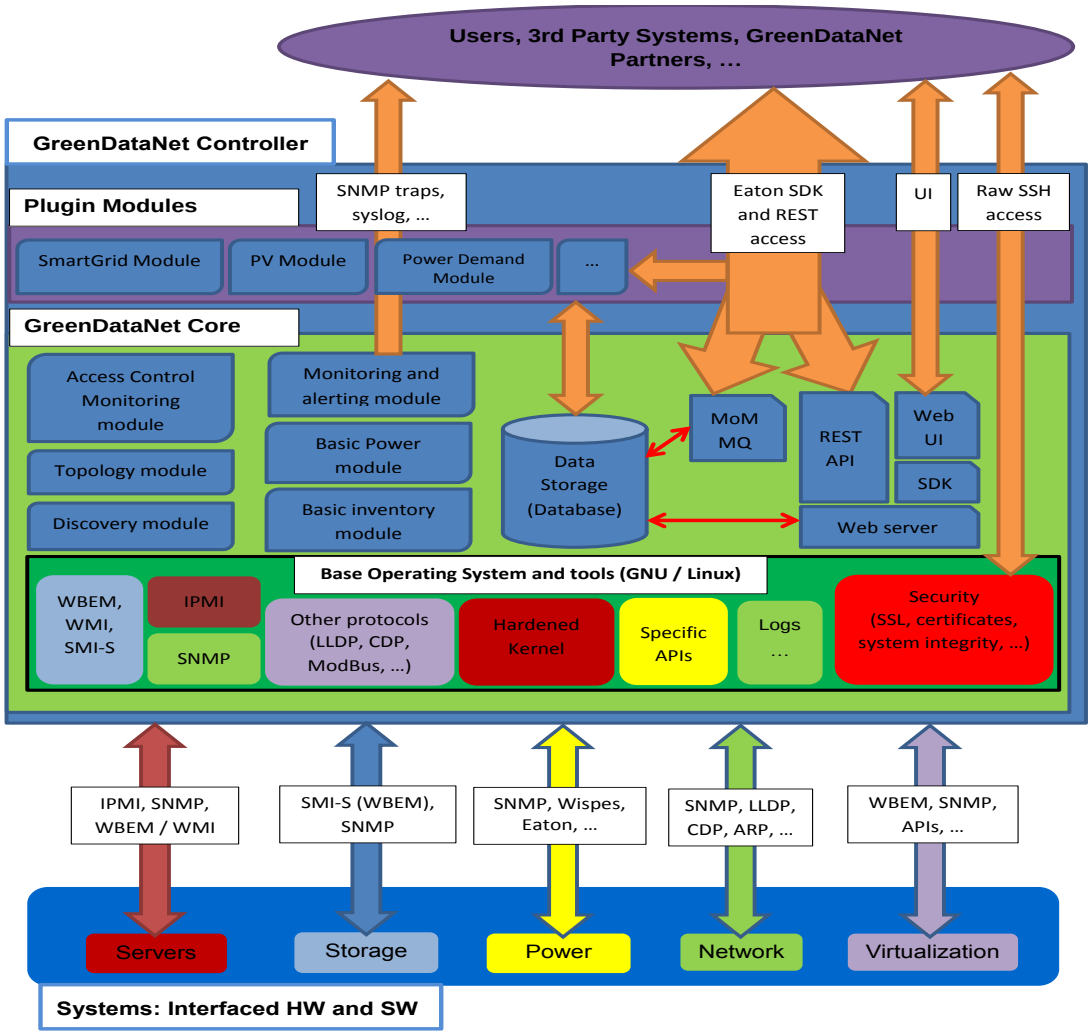
- Scalability: to be able to aggregate large amounts of data provided by each of the levels above, whatever the size of the datacenter,
- Adaptability: to be able to provide the required modules, to communicate with the various IT systems, either SW or HW, found in the specific datacentres.

To satisfy these requirements, GreenDataNet will leverage the concept of "**Boxes"**, made by Eaton.

These GreenDataNet boxes are IT appliances, which system architecture is a combination of several components, or modules, running over a tailored and secured operating system. It runs on industrial hardware (physical appliance) or in a virtual machine (virtual appliance).

4.2. GENERAL ARCHITECTURE

The following illustration provides a high-level visibility of GreenDataNet system architecture, describing the main functional domains and notable architecture points:



4.3. ARCHITECTURE MODULARITY

In order to gain flexibility and future extensibility, along with being able to address variations of datacentres sizes and requirements, the present architecture relies on 3 main pieces:

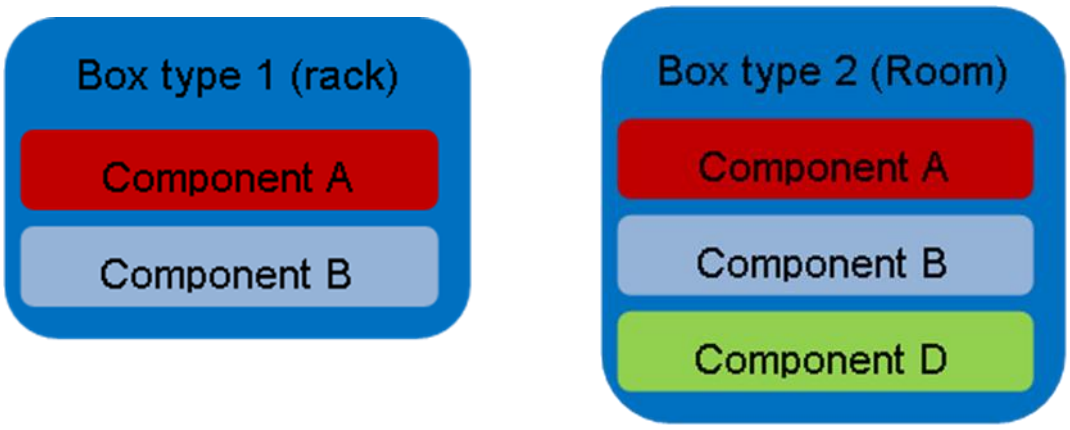
- A base operating system, providing basic features to run SW, interface with various protocols and provide self-update and self-healing capabilities
- An appliance core, providing generic modules for requirements such as security, communication interfaces, data storage and persistence
- A functional core layer, providing a "plugin layer" for specific modules, such as an Energy Management module with specific Solar PV considerations or expert components provided by GDN partners.

These modules, or components, whatever the layer these belong to, will be providing a set of features, suitable to address the specific needs of a datacentre related to:

- interfacing with the monitored systems (devices and SW)
- providing functional features, such as the calculation of metrics and communication interfaces with 3rd party systems.

As illustrated below, each module (or component):

- Can be deployed on HW and / or SW boxes,
- Must be reusable across the different GreenDataNet use cases,
- Must leverage the "Openness" factor (see §4.6)



4.4. GREENDATANET APPLIANCES

These GreenDataNet boxes are IT appliances, the system architecture is a combination of several components, or modules, running over a tailored and secured operating system. It runs on industrial hardware (physical appliance), but may also be deployed as virtual machine (virtual appliance).

Throughout the present document, different terms will refer to the same GreenDataNet Appliance:

- GreenDataNet box,
- GreenDataNet Controller, or GC,
- Rack Controller, or RC.

4.5. TOPOLOGIES AND URBAN DATA CENTRE TYPES

Depending on the data centre size and complexity, which relates to the Type (I or II) of Urban Data centre, one to several GreenDataNet appliances will be needed.

DC of Type 1 (up to 12 racks) will basically require 1 GreenDataNet controller.
DC of Type 2 (up to 80 racks) multiple controllers, with 1 master (main controller) and the other acting as "slaves" (secondary controllers)

4.6. OPENNESS CONSIDERATION

A key criteria of the GreenDataNet architecture, beside from the flexibility and extensibility, lies in its openness nature.

This point represents the capability of this architecture to communicate and collaborate with 3rd party modules.

To that end, open communication interfaces will be available, leveraging the state-of-the-art web technologies such as REST. Such communication interfaces provide an easy way to connect remote SW components, to exchange information and provide more features.

Moreover, beside from open communication interfaces, GreenDataNet will also maximize the use and production of:

- Opensource software projects,
- Open Standards and communication interfaces

5. GREENDATANET SYSTEM CONSIDERATIONS

5.1. APPLIANCE OPERATING SYSTEM CONSIDERATIONS

GreenDataNet selected **Emdebian Grip** as the appliance Operating system.
Emdebian Grip is a small-footprint Linux distribution, based on and compatible with Debian GNU/Linux, intended for embedded systems.

5.2. GREENDATANET STARTUP PROCESS

This chapter will describe the specific aspect during GreenDataNet system initialization.

5.3. SECURITY

In modern architecture, security becomes more and more important. Security is one of most important key feature of GreenDataNet appliance.

Several security aspects are described here: integrity, user authentication and privacy.

The GreenDataNet appliance monitors file integrity (FIM) during system initialization.

GreenDataNet File integrity monitoring (FIM) is an internal control or process that performs the act of validating the integrity of operating system and application software files using a verification method between the current file state and the known.

Proposition: In GreenDataNet appliance, we rely on a lightweight **OSSEC** agent that is provisioned, managed, and monitored via our web-based console. As soon as any changes are executed, an alarm is triggered in our Alarms window. These might be changes that do not require a response; however, it's important to monitor all activity to capture baselines, and notice any abnormalities like policy violations or potential system compromise. Since GreenDataNet OS will be Linux, we may use the iNotify technology, to get asynchronous notifications (events) when interesting files are modified.

Second aspect is user authentication, in GreenDataNet appliance; authentication is based on Linux Pluggable Authentication Modules (PAM) which are a common framework for authentication and security.

As a typical example, GreenDataNet may support Active Directory (2008) windows domain controller integration. For this, GreenDataNet appliance includes Kerberos v5 (pam_krb5) and GSS-API client libraries for SASL authentication.

The last aspect is the privacy; it is planed that GreenDataNet will support SSL communication (TLS1.1).

5.4. HARDENING

GreenDataNet appliance comprises a linux kernel with suitable security protections, such as:

- PAE (Physical Address Extension) support,
- ASLR (Address space Layout Randomization) support,
- NX (Non-executable) memory protection support.

There are also numbers of Userspace (system) protections, such as Firewall setup, symlink and hardlink Protection, ptrace Protection, … that will be enabled, to ensure a high level of security protection of the GreenDataNet system.

5.5. GREENDATANET CORE SYSTEM SERVICE

During the system initialization, GreenDataNet appliance starts several services, that are called GreenDataNet core system services.

As an example, the logger, the data base engine, the message bus, the module manager, and the web server are started as daemon process.

Start, stop and status of GreenDataNet core system services are defined in /etc/init.d directory, or similar, depending on the exact init system used (sysV, systemd, …).

5.6. HOSTNAME, DNS AND MDNS PUBLICATION

After system initialization, the GreenDataNet appliance declares itself as "`gdn-XXYYZZ`" hostname, where `XXYYZZ` are the last 6 characters of the MAC address of NIC eth0.

For example: `gdn-0272be`

To achieve a zero configuration host name resolution service goal, GreenDataNet uses multicast Domain Name System (mDNS). It uses essentially the same programming interfaces, packet formats and operating semantics as the unicast Domain Name System (DNS) to resolve host names to IP addresses within small networks that do not include a local name server.

The mDNS protocol is published as *RFC 6762*, uses IP multicast User Datagram Protocol (UDP) packets, and is implemented by the Apple Bonjour and Linux Avahi and nss-mdns services.

GreenDataNet choose Apple **mDNSResponder** under Apache Public License as mDNS implementation.

mDNSResponder allows to publish extra attributes, such as:

- `GC-SerialNum` : STRING, the last 6 characters of the MAC address
- `GC-Type` : INTEGER From 1 to 5, depending on the type (resources) of the appliance
- `GC-Role`: STRING {master, client}. After booting, GC-Role=client

How does a GreenDataNet appliance decides to become master ?
An mDNS request (dns-sd) gives the list of all GDN appliances known on the network; each appliances is able to sort this list by GC-Type and GC-SerialNum. The appliance which gets the maximum (GC-Type) and max(GC-SerialNum) becomes the master. It then publishes its new role GC-Role=master attribute and change its hostname from "***GC-XXYYZZ***" to "***GC-master***".

5.7. TIME SYNCHRONISATION

GreenDataNet appliance supports NTP v3 and v4 (RFC5905, RFC5906, RFC5907).

It is important to note that NTP needs to be configured before integrating GreenDataNet appliances into Active Directory domain, and before starting the .

5.8. SOFTWARE PACKAGING

GreenDataNet core and additional modules will be provided as software packages, including the actual software and meta-data to describe it. This system allow for a greater user experience, along with fine-grained software update.

The preferred and default GreenDataNet packaging is the Emdebian Grip package, also known as *Debian package* or *.deb*.

GreenDataNet appliance may be jailbreaked to get the shell prompt and customize it. By this way, it is possible to install some functional module that are not part of the official distribution.

5.9. SYSTEM LOGS

Rsyslog and logrotate manage GreenDataNet system logs. Details on the information logged (severity, format, …) along with the retention and rotation times are still to be completed.

6. SERVICE ORIENTED ARCHITECTURE AND MESSAGE-ORIENTED-MIDDLEWARE

6.1. INTRODUCTION

The main idea in the software architecture of GreenDataNet is to define something where business layer is implemented as service (we call them Module after) where services will be loosely coupled, autonomous and reusable. This is known as Service-Oriented-Architecture. This approach requires a message bus as a central bloc of the GreenDataNet architecture.

A Message-Oriented-Middleware (MoM) is a software infrastructure supporting sending and receiving messages between distributed systems, either synchronously (request – reply pattern) or asynchronously (publish – subscribe pattern). A MOM allows GreenDataNet application modules to be distributed over heterogeneous platforms (rack controller or not) and reduces the complexity of developing, while improving the flexibility and extensibility of the software architecture.

GreenDataNet selected AMQP 1.0 as the MoM technology, since is an official standard (ISO/IEC 19464), open and secured.

The Advanced Message Queuing Protocol (AMQP) defines an agnostic message protocol formats used between participating application components, so implementations are interoperable. AMQP may be used with flexible routing schemes, including common messaging paradigms like point-to-point, fan-out, publish/subscribe, and request-response.

6.2. DISTRIBUTED ARCHITECTURE AND FEDERATION CONSIDERATION

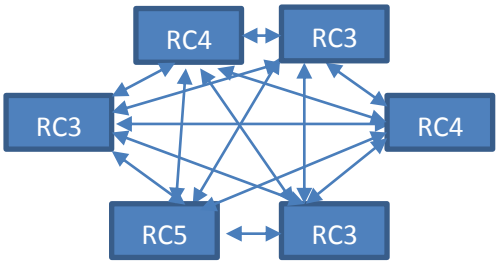
To support the requirements related to urban data centres of Type II, along with networks of data centres, GreenDataNet must provide a distributed architecture, capable of self-adaptation, to federate the various controllers that are deployed in data centres. Moreover, the networks of data centres will require even more federation capabilities.

Dynamic communication routes allow a client to create bindings to an exchange on one broker, and receive messages that satisfy the conditions of these bindings not only from the exchange to which the client created the binding, but also from other exchanges that are connected to it using dynamic exchange routes.

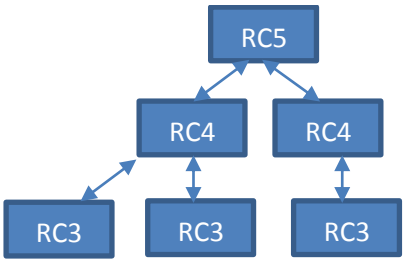
Combined with mDNS publication feature, a GC is able to detect other GC and so able to add dynamic routes to interconnect themselves to configure a federate network.

A federated network is generally a tree, star, or line, using bidirectional links (implemented as a pair of unidirectional links) between any two brokers.

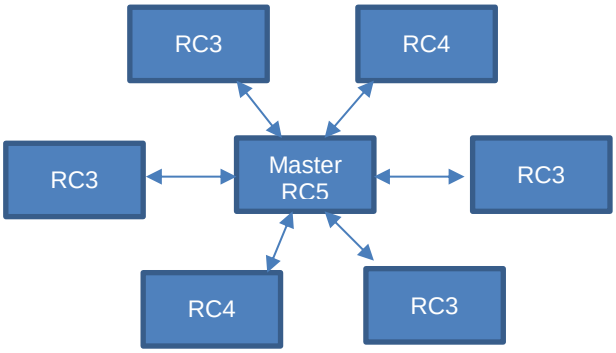
In the case of a distributed architecture with several GC installed. The lazy approach is to interconnect them into a start network where every GC is connected to all other GC. This architecture may generate a huge overhead bandwidth.



Another approach is to configure a federated network as a tree. It introduces some complex algorithm to allow each GC decides to which GC is need to be connected. This approach seems to run in a static architecture and may meet some issues when the topology changes.



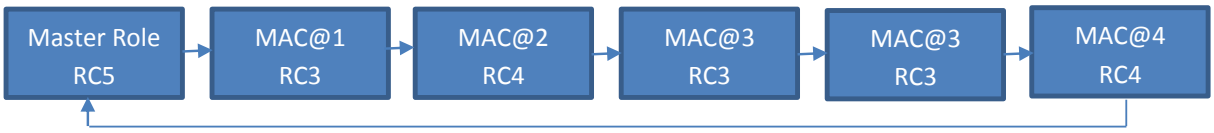
A star network with Master node as central node seems quite simple to configure.



A ring topology is also possible. It is simple, efficient and self-adaptive. In such design, only unidirectional links are used.

As each GC hostname includes a part of its MAC address, all hostname are unique. A mDNS request (dns-sd) gives the list of all GC known on the network, each GC is able to ordering this list by GDN-SerialNum attribute (6 last digits of MAC address) and add dynamic routes to the next closest RC.

The first and the last node are specific: The first node is always the Master Role GC (or the smallest GC-SerialNum if master is not yet elected) and the last node must add dynamic routes to the master node.



At this level of the project, we will focus more on the third solution where each GC client (GC-Role) is connected to one master node. This decision is planning-driven.

Note: a dispatch router is also another alternative, available in recent Qpid project but interesting to follow it.

6.3. BROKER AND CLIENT IMPLEMENTATION

GreenDataNet is currently evaluation various MoM solutions, such as:

- *Qpid C++* project which is the 100 % compatible with the Advanced Message Queuing Protocol Standard,
- *Proton*, a subproject of Qpid, is a lightweight implementation of the AMQP protocol. Using proton it is likewise possible to develop clients and servers,

- *ZeroMQ*, and its related projects such as ZCCP, is a lightweight MoM, developed by a member of AMQP committee. It is however not compatible with AMQP, and may thus be used as a lightweight internal bus, to allow communication between modules, inside of a GreenDataNet Controller.

7. GREENDATANET MODULES

7.1. INTRODUCTION

Modules are where all the work takes place. There are modules for doing discovery, asset, monitoring, correlation, optimization, etc.

7.2. DESCRIPTION

The modules are all separate processes that can be started from the module manager program or run by the user from the command line.

The module to serve messaging protocol uses MOM and therefore can communicate over a network. This should give the system the required scalability capabilities.

Most of the startup work would be creating compound data types and tags. Once this is done the module signals the server that it is "Running."

Several languages are considered to implement GreenDataNet modules:

Language	GC	Third party
C	✓	✓
C++	✓	✓
LUA (1)	✓	
Python	? (2)	✓
Java		? (2)

(1) LUA may be used as internal RC scripting language. It may be considered only as a plugin scripting language to extend some module features.

(2) need to check HW capacity.

An interesting approach is to define modules as MoM agents: each agent provides a schema that describes its management model including the object classes, methods, events, etc. There are two drawbacks, the first one is that agent framework is coupled with the MOM technology, for example QMF for QPid Bus. The second drawback is how it can work on smaller GC where probably no Broker will be present.

Configuration file format, directory (module tree) organization will later be described here.

7.3. GREENDATANET MODULE MANAGER

The features of the module manager are to manage the lifecycles of the modules that are configured.

The modules manager thus handles the starting, stopping and restarting (or reloading) of these modules.

Modules may be started in a specific order, to handle inter modules dependencies.

It is also possible to group the modules into startup groups. Once it has started modules it waits until each one report that it is "Running." Once they are all running it moves on to the next group.

Controlling the order of module start up allows manage module dependency.

Module List => see QMF CLI tools

7.4. CORE MODULES

7.4.1. INTRODUCTION

Core modules are really the basic and mandatory modules to get a base GreenDataNet appliance working.

To understand what core modules are, it is important to present what the main data flow of GreenDataNet appliance is:

The figure bellow will be detailed in a future version of this document.

It is worth to note that:

- *Maintain RT values* also comprise data processing such as metrics calculation
- *Display* only emphasizes on graphical user interface
- *Action* comprises everything that is not simple data post-processing (such as metrics calculation) nor user interface display. Features such as *Optimisation* belongs to this category.

7.4.2. DISCOVERY MODULE

This module is in charge of discovering the systems to be monitored.

Discovery module starts a new scan in following situation:

- After Discovery module initialization, a full scan is done
- When a new sub-net is added or modified
- Through CLI command. Some filter may configured to choose option like full-scan, specific sub-net, specific device
- Through REST API command. Some filter may also be configured
- On a time-base (every X minutes for example)

When module discoveries a new network equipment, its local Data Base discovery table is updated, and a message is send on topic discovery.new on the bus to notify master node and other third party modules.

In a distributed architecture, several GC may discover the same equipment. In such situations, the master node decides which GC will be in charge of inventory and monitoring it. The other GC may thus be used as a fail-over system, in case the primary controller is not available.

The main criteria are:

- The network distance between a GC and the equipment (also known as “network hop”). The closest GC has a higher priority,
- The type of GC. A smallest GC has more priority, since it is theoretically nearer to the monitored devices,
- The system load of the GC.
- In case of issue on the primary RC node, a second RC node (failover node) is selected to do inventory and monitoring job.

For this feature, the master node stores all discovery messages data in a Meta discovery table to know which GC has discovered what and when. The master node notifies client nodes each time a change appear.

7.4.3. INVENTORY AND ASSET MODULE

This module is in charge of collecting identification and processing topological data, to build topology maps (Basic power cabling status, Basic network cabling status, relationships between IT loads, such as Servers and Storages).

Asset module starts a new identification job in following situation:

- After Asset module initialization, checks all discovered, affected by master node and non-inventoried devices declared is its local Data Base Discovery table to do inventory.
- On equipment/RC affectation message reception from master node.

The asset module supports import feature for provisioning equipment and credential, it could be done by CLI command line or REST API.

7.4.4. MONITORING MODULES

These modules are in charge of:

- Collecting the devices data and health status (Key parameters and Critical alarms), through the communication interfaces described hereafter in the Protocols stacks chapter,
- Send collected data to real time data module.

7.4.5. SERIAL MODULES

Serial modules are specific subtypes of monitoring, they manage point-to-point serial port device.

Serial modules configuration are done manually because it is a little bit complex to do automatic discovery,

This include devices using RS232 and RS485 communication, along with simple dry-contacts system (potentially using GPIO ports), such as sensors and probes.

This does not include USB and network communication interfaces, which can be enumerated (smart discovery of capabilities).

7.4.6. REALTIME AND HISTORIAN STORAGE DATA MODULES

This module is composed of two major parts:

- the real time data module
- the historian data module

The real time data module is composed of a data map where the live-data values are stored. Its goals are:

- Support real time response time GreenDataNet feature (latency to be defined),
- Support notification mechanism, when value changes beyond a given threshold,
- Aggregate data:
    - A data built from other data
    - A consolidated / compiled / reduced data
- Send events and alerts, according to status and configuration
    - Through MoM, for GreenDataNet dashboard and other GreenDataNet plugins
    - Through syslog, REST interface and SNMP traps, for 3rd Party Systems integration
- Implement current value request API.
- The data map is allowed to grow as the number of tags increase.

The real time module could be a simple in-memory hashmap table or a more sophisticate like **redis** open source project. But it could design as a CIM server (thinking still in progress).

The second part is the historian data module. Its goals are:

- Encapsulate the data, it stores data in persistent way . Filters likes Deadband may be available to store complete or partial value.
- Replicate (mirror) data on other storage node to get redundancy, ease of use, availability (including resilience) or performance.
- Implement historian value request API,
- Aggregate data (Compute trends,)
- Archive old data and remove them from on-line historian DB.

The historian storage data module may be different depending on kind of platform:

- RC1-RC2 support only simple flat file with small retention delay
- RC3-RC5 support relational Data Base

When replication is activated, the historian module push modified data on a specific topic which is routed to other RC. In this version, only RC-master is notified.

With this architecture the live data and the static data are separated.

Each other module would send messages to the storage module to read and/or write data.

7.5. FUNCTIONAL MODULES

7.5.1. POWER MODULE

This module will provide basic power features, such as Power usage reporting and Power load extrapolation. This module may include the SmartGrid interface too, otherwise it will be subject to a dedicated module.

7.5.2. OPTIMISATION MODULE

This module will provide the engine and policy rules to apply optimization to the data centre. This potentially includes a bridge to the SmartGrid.

7.5.3. OTHER MODULES

Some other modules are still under consideration, and may be detailed later.

7.6. THIRD PARTY MODULES

To support the openness requirements, and allow third parties to develop modules that interact with GreenDataNet, some communication interfaces will be available.

These communication interfaces are detailed in §10.

7.7. PROTOCOLS STACKS

This chapter is summarize the various communication protocols available for GreenDataNet Core and functional modules development.

7.7.1. WBEM, WMI , SMI-S

This module (library and command-line tools) provides the capability to exchange data over WBEM, and derived standards such as WMI (WBEM implementation on Windows systems) and SMI-S (WBEM implementation for storage systems).

7.7.2. IPMI

This module (library and command-line tools) provides the capability to exchange data over IPMI.

This will be used to interface with servers, to get data and send OOB orders (start a server remotely for example).

GreenDataNet appliance has selected **FreeIPMI**. FreeIPMI provides in-band and out-of-band IPMI software based on the IPMI v1.5/2.0 specification.

Equivalent proprietary protocols, such as HP iLO and Sun / Oracle iLOM / ALOM may also be considered, as a future improvement.

7.7.3. SNMP

This module (library and command-line tools) provides the capability to exchange data over SNMP.

This will be used to interface with devices from various categories, such as servers, storages, network and power devices, along with some SW systems.

GreenDataNet appliance uses **Net-SNMP**. Net-SNMP is a suite of applications used to implement SNMP v1, SNMP v2c and SNMP v3 using both IPv4 and IPv6.

7.7.4. OTHER PROTOCOLS (LLDP, CDP, MODBUS, …)

This module compiles a number of other protocols, that will support specific devices, and complement data gathered through the main communication interfaces.

Among these protocols, we can cite:

- ModBus and OpenMBus, to interface with facility systems (such as cooling), and industrials systems (such as PLC), both in serial (RTU) and network (TCP) modes,

- LLDP / CDP, which will support the discovery of the network topology. Note that some of these information will have already been obtained through SNMP.

7.7.5. NUT

Open source project **NUT – Network UPS Tools** is used in GreenDataNet, to interface with the many UPS, ePDU and other power devices supported.

8. GREENDATANET DATA MODEL

8.1. INTRODUCTION

GreenDataNet systems implement a distributed database, commonly referred to as a *Tag Database*, which contains live data elements. It is composed of two parts: the real time and the historian part.

Historian data and all other "non-live" data are managed by the same DB engine.

8.2. REAL TIME

Real time are provided to address performance issues. It can be considered as a cache of the historian DB.

Tags are the current in-memory key:value pair.

Tags are configured at startup time. Each GreenDataNet module declares its own tags list .

Tag attributes are:

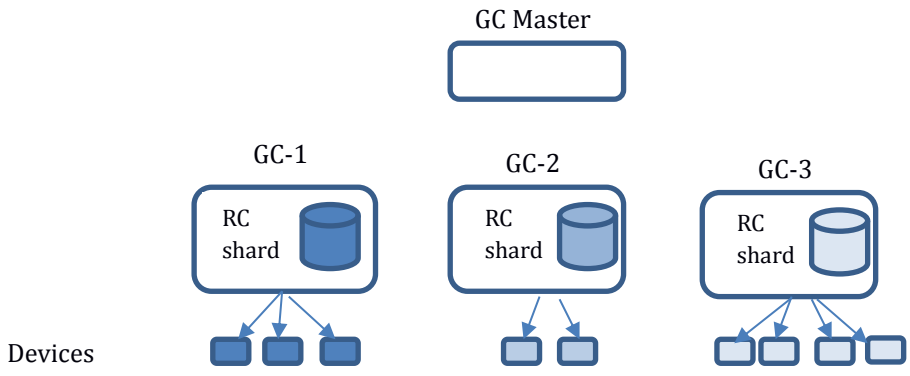
- Tag name
- Type ?
- Historian policies (deadband, retention duration)

8.3. DISTRIBUTED HISTORIAN ASPECT

This chapter describes two main aspects of the organization of data distribution in GreenDataNet: the shared database and replication.

The GreenDataNet historian database architecture is designed as a shard database; a shard is a horizontal partition in a database or search engine. Each individual partition is referred to as a RC shard or RC database shard.

In the figure below, each RC3 is responsible for storing historian data from a subset of devices. The master node doesn't have any historian data from field devices. If a request is done to master node, the master send a request on a specific MoM topic, only RC-client which have the answer reply on the bus.

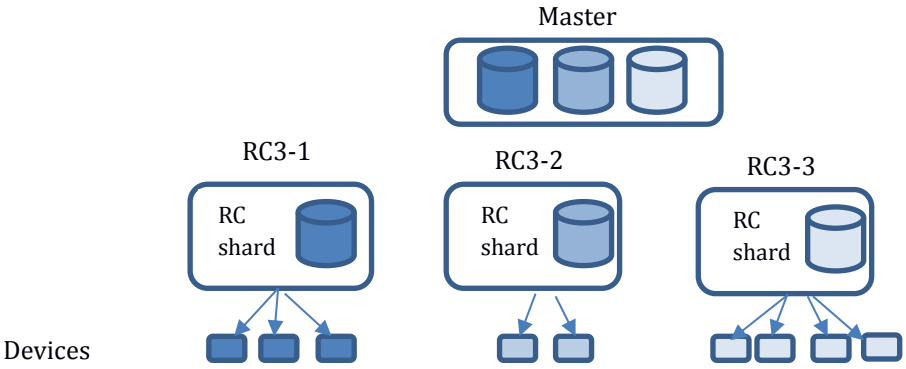


From an availability point of view, data should be replicated. The number of replica (or level of redundancy) is configurable in GreenDataNet:

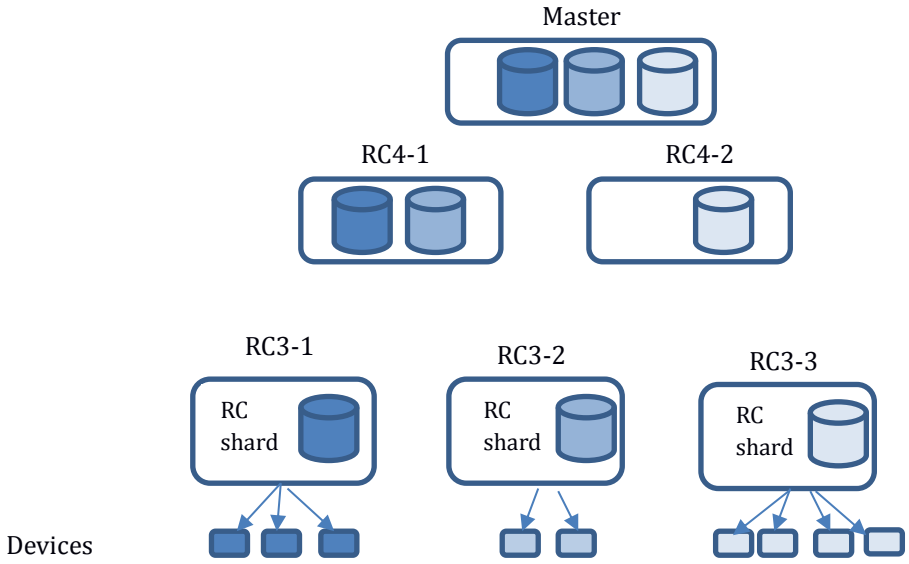
- None
- One (high availability)

- Two (very high availability)

In the current version of GreenDataNet, the number of replica could be none (0) or one. If one replica is activated, all historian data are replicated to the master node (see below figure).



In a future release of GreenDataNet, the replication process may be more sophisticated where the master node may spread replication among several RC client nodes. The figure below shows an example of a very high availability replication level implementation (replica=2).



8.4. DATABASE ENGINE

The current approach on GreenDataNet Database engine is to suggest two types of DB engines:

- A relational open source Data Base likes **PostgreSQL**, **MySQL** or **MariaDB** on higher-end GreenDataNet controllers platform,
- **SQLite** engine on lower rack controller.

The currently selected database engine is MariaDB.

8.5. DATABASE STRUCTURE

This chapter presents the considerations related to the database.

It will present both a logical model (sort of functional specification for DB) along with a physical model (sort of implementation).

Both are currently under completion, but the main guidelines are the following:

- Devices
    - Identification / Endpoints
        - Shared referential between heterogeneous system
    - Device Data / Measures / Alarms
    - Logs
    - Settings
- Groups / Infrastructure / Topologies
    - Description
    - Strategies / Policies (Failover / Redundancy / Load Shedding / ...)
    - Infrastructure Connectors (Connectivity / Roles)
- Settings Management
    - Events / Actions
    - Views
    - Acquisition
- Access Rules / Right Management

9. GREENDATANET USER INTERFACES

9.1. INTRODUCTION

GreenDataNet users access rack controller infrastructure through a web interface. The web interface complies with all state-of-the-art principles of "web 2.0" user interface.

9.2. APPLIANCE WEB SERVER

GreenDataNet Controller Web Server provides several central features. It must deliver static resources, allows managing security / user authentication and delivers business interface, such as REST API over HTTP.

In this version security and user authentication is not yet available.

For more detail about REST API, see chapter 10.3.

In a distributed architecture, the master GC is the preferred user entry point. As master node records itself in DNS server as "***GC-master***". Thus, the web interface is available on ***http://GC-master*** URL.

So far, GreenDataNet has selected the ***lighttpd*** open-source web server, since it is optimized for speed-critical and resources constrained environments. However, the versatile approach selected for GreenDataNet allows for an easy replacement of such components. Thus, lighttpd may easily be replaced by some other component, such a NGINX.

Lighttpd supports the FastCGI, SCGI and CGI interfaces to external programs. REST API is implemented based on one of those interfaces. It is interesting to note that lighttpd's FastCGI may be configured to support PHP.

Lighttpd modules used are: mod_alias, mod_access, mod_rewrite, mod_expire, mod_fast_cgi.

9.3. WEB CLIENT

The web client runs on modern web browser.

The web client support several languages, the supported languages may be listed here.

The supported browsers are:

- Internet Explorer 8+

- Firefox X.X

- Google Chrome

The previous list needs to be confirmed by web UI team.

From an architecture point of view, the web client design is HTML5.0 oriented; it is based on modern Javascript framework and CSS (at least 3). The architecture is really RESTFul oriented, all business dynamic data are retrieved by web client through the REST API in an asynchronous way.

10.GREENDATANET COMMUNICATION INTERFACES

10.1. INTRODUCTION

GreenDataNet appliance supports:

- Command-line-interface (CLI) administration interface, by remotely connecting to the controller through SSH
- REST API interface
- AMQP interface

10.2. COMMAND LINE INTERFACE

Administrator of GreenDataNet appliance uses a shell environment with limited "functional" GreenDataNet command.

GreenDataNet appliance is considering the use of the opensource **clish** project. This framework implements CISCO-like CLI. GreenDataNet commands are defined using XML files.

Commands will be described in administration manual.

10.3. REST JSON API

Technical solution for REST server should be selected.

The REST server may interact with GreenDataNet modules throw the MOM. The REST server may also connect to the GreenDataNet Data Base.

The solution considered is a c++ REST implementation, with a framework similar to restcgi and proton.

10.4. MOM API

This point will be detailed later.
The complexity of using a such API will be addressed by a SDK – Software Development Kit, that will make it easy to interface with GreenDataNet, and get or provide data to the system.

11.CONCLUSION

The software architecture of the data center energy management system described in this document relies on hardware appliances to ease acceptance of end users who are frequently reluctant using “big software” solutions.

As described in chapter 4 the solution proposed may allow to interface with:

- existing protocols (like IPMI) and future ones using in band network,
- existing solutions and new ones using out band network as for the PMSM solution (investigated by EPFL and Credit Suisse) by adding a wireless sensor network (IEEE 802.15.4) with a USB dongle.

In both cases this will allow to fuel our data base with data to run optimization algorithms.